Example-based Volume Illustrations

Aidong Lu
Purdue University & UNC Charlotte

David S. Ebert*
Purdue University

ABSTRACT

Scientific illustrations use accepted conventions and methodologies to effectively convey object properties and improve our understanding. We present a method to illustrate volume datasets by emulating example illustrations. As with technical illustrations, our volume illustrations more clearly delineate objects, enrich details, and artistically visualize volume datasets. For both color and scalar 3D volumes, we have developed an automatic color transfer method based on the clustering and similarities in the example illustrations and volume sources. As an extension to 2D Wang Tiles, we provide a new, general texture synthesis method for Wang Cubes that solves the edge discontinuity problem. We have developed a 2D illustrative slice viewer and a GPU-based direct volume rendering system that uses these non-periodic 3D textures to generate illustrative results similar to the 2D examples. Both applications simulate scientific illustrations to provide more information than the original data and visualize objects more effectively, while only requiring simple user interaction.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—color, shading, and texture

Keywords: Volume Illustration, Example-based Rendering, Wang Cubes, Texture Synthesis, Color Transfer.

1 INTRODUCTION

Scientific illustrations play an essential role in education and training. For example, illustrations are a vital medium in teaching anatomy, explaining biological processes, highlighting anomalies, and explaining surgical procedures. They are commonly used as examples to explain structures (e.g., shape, size, appearance, etc.) and provide stylized or additional information to acquired datasets (e.g., CT and MRI). Therefore, exploring an example-based rendering method can adapt traditional illustration techniques to more effectively present information and provide a familiar environment to those who have been trained with similar images for years.

Because of the inevitable information loss during the acquisition of scientific datasets, many illustrators employ textures to enrich object details. These textures usually provide more information beyond the original data resolution and improve the understanding of the real objects. For example, Figure 1 shows an illustration of a slice through the upper abdomen and a corresponding magnetic resonance image (MRI) [27]. The illustrative section provides much more information than the MRI by adding fine structural detail and clearly delineating organs with accepted conventions, such as drawing the veins in blue and arteries in red [13].

Although scientific illustrations do not exactly replicate real subjects, the illustrators follow specific methodologies and procedures to concisely, accurately, and effectively *convey* the important aspects of the subject, such as shape, location, orientation, and structure. Figure 2 shows two pairs of scientific illustrations and high-resolution medical images from the Visible Human Project [1] comparing the same organs of the human body. These examples demonstrate a strong similarity between scientific illustrations and the real subjects. They also show that illustrators usually modify and/or simplify the size or position of a real subject to achieve a coherent structure, and change real colors to distinguish one object from the surroundings and improve understanding. For example, in Figure 2(a), the geometry and detail of the spinal disk (in blue) has been simplified since it is not the focus of the illustration. On the whole, these examples demonstrate the strong ability of scientific illustrations to provide expressive and additional information compared to medical images. Similar to scientific illustrations, the utilization of textures can provide more information and serve as an additional method to distinguish different objects, matching the visualization objective of feature rendering and exploration. However, the large memory requirements and the need for both high-resolution 3D texture synthesis methods and tedious user interaction limit the rendering features, resolutions, and styles.

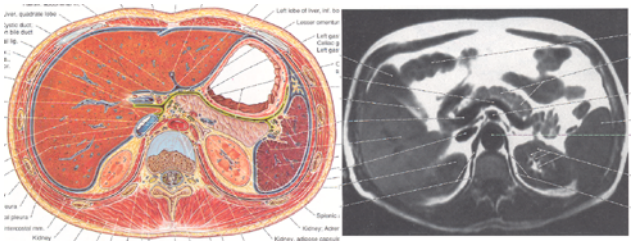

Figure 1: A scientific illustration shows more general information with colored textures than the corresponding MRI.

In this paper, we present a method to generate illustrative renderings of volume datasets using example illustrations and photographs. Our method is composed of a texture synthesis and color transfer method to generate illustrative 3D textures. Once the illustrative textures are generated, volume datasets of similar subject matter can be interactively rendered using any of these illustrative textures. As in scientific illustrations, these 3D textures can be used in volume applications to enrich detail and achieve illustrative and artistic rendering styles. Specifically, to emulate example 2D illustrations, the 3D textures are automatically generated by recoloring available 3D samples using color distributions. Wang Cubes [5] are then synthesized from the colored 3D samples and used to generate non-periodic volumetric textures. The textures for all the materials/objects are efficiently mapped into a volume, enriching both detail and the rendering, while using a small amount of texture space.

For volume illustration, volume textures of real or similar objects are chosen as input to provide meaningful information. First, their colors are transferred using selected sample illustrations to generate illustrative textures (Section 3). Next, the illustrative textures are used to synthesize a set of Wang Cubes (Section 4). These synthesized Wang Cubes can tile any sized volume texture and are used in interactive rendering (Section 5). To match object features, we generate one texture for every object in the volume. The users are only required to provide the input textures and illustration examples prior to the interactive rendering.

We have developed two applications using these new methods.

*e-mail: {alu,ebertd}@purdue.edu

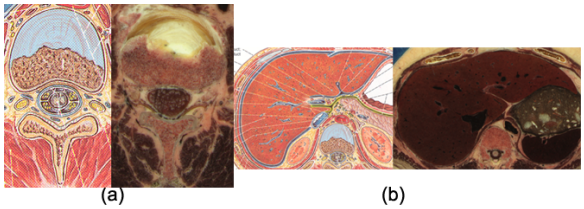

Figure 2: The comparison of scientific illustrations (left, which are enlarged from Figure 1) and high-resolution medical images from the visible woman (right) around the vertebra and liver. The images are scaled to match each other.

Our real-time 2D slice viewer allows users to create continuous illustrative slice views for any cut direction through the volume, while our GPU-based direct volume renderer generates volumetric dataset illustrations, where the illustration styles and effects can be interactively modified through multiple transfer functions. Both applications take advantage of the abilities of scientific illustrations to effectively convey object properties, and save tedious user interaction through our new example-based texture generation and color transfer methods.

2 Related Work

Volume illustration techniques take advantage of the effectiveness of scientific and artistic illustrations to improve volume visualizations in many aspects. For instance, Kirby et al. [15] combined multiple data values to generate 2D flow images by utilizing concepts from paintings. Saito explored the usage of simple primitives on isosurfaces to depict volume shapes [23]. Owada et al. [20] presented an approach for polygon meshes using texture synthesis on a surface. Here we concentrate on general illustrations of volume datasets, extending the early work of combining non-photorealistic rendering (NPR) and volume rendering techniques to enhance important features and regions [7]. Artistic rendering effects were achieved by implementing artistic procedures and simulating illustration techniques [18, 28]. Recently, these techniques have been extended with hardware-accelerated rendering techniques to render large datasets and produce high quality results [9, 17].

Wang Tiles [29, 30] are used to generate non-periodic textures in computer graphics by following their placing rules. Analogous to Wang Tiles, Culik and Kari [5] introduced Wang Cubes with colored faces. Several researchers have used Wang Tiles and Wang Cubes to generate textures and patterns in computer graphics, including Jos Stam [26] for water simulation, Neyret and Cani [19] for stochastic surface tiling, Cohen et al. [4] for non-periodic image generation, Sibley et al. [24] for video synthesis and geometry placement, and Lu et al. [16] for illustrative volume rendering using geometric primitives.

Example-based approaches have been used in many fields, such as machine translation, image processing, and rendering. Because of their ability to "learn" from examples, these approaches take advantage of useful information that is difficult to summarize, and simplify user interaction. Within this field, we are especially interested in two categories, example-based rendering and texture synthesis. Example-based rendering generates new results by simulating the drawing styles of examples. Freeman et al. [8] used Markov random fields to learn the transformation from captured image to scene interpretation. Hertzmann et al. [12] presented the image analogy framework. Drori et al. [6] and Hamel and Strothotte [10] generated new images by extrapolating multiple example style fragments. The example-based idea has also been explored for texture synthesis methods in various applications, including the work of Haro and Essa [11] and Chen et al. [3]. Also, Wang and Mueller [31] generated missing details from available high-resolution data for virtual microscopy by matching the data source details on multiple levels.

3 Color Transfer

3.1 Problem and Assumptions

To simulate arbitrary illustration examples for volume applications, we must solve the problem of synthesizing high resolution three-dimensional textures from two-dimensional illustration examples. Achieving high quality 3D textures from 2D examples is a difficult task [32]. Another limiting factor is that many texture examples have insufficient resolution for texture synthesis since they are cropped from illustrations and bounded by the size of the real objects. On the other hand, these small texture examples provide useful color distributions. Since scientific illustrations usually represent important, real object features, it is common to see strong similarities between the illustrations and the real objects, and between two illustrations of the same objects even when they use different drawing styles (Figure 2). We can use these similarities to change the problem of three-dimensional texture synthesis to a color transfer problem by using available 3D source textures (e.g., color section volumes, MR, CT scalar volumes).

To simulate the styles of example illustrations, we transfer both the chromatic (two channels) and luminance values from the illustrations to the source dataset. Since a 3D texture is treated as an array of color (or grey-scale) information, the color transfer problem has no fundamental differences from color transfer between 2D images. Our work is based on Reinhard's method [21], which transfers colors between colored images by mapping the mean and standard deviations along each color channel and uses distance-weighted colors from separate swatches to improve the results, since the quality depends on the composition similarity between the example and source images. To save some color blending, Welsh et al. [33] transfer colors between corresponding swatches and they target transferring colors to grey scale images by matching luminance and texture information. The effectiveness of the result is highly dependent on the user's selection of corresponding regions in each image. However, the user interaction required by their method to adjust the corresponding regions manually becomes much more difficult and tedious for 3D textures. Therefore, we present a fully automatic color transfer method for 3D volumes from user-provided example photographs or illustrations to source data volume.

In medicine and many biological fields, photographic volumes for many subjects are readily becoming available (e.g., The Visible Human Project). Our technique works for transferring color from 2D sources to three-dimensional volume datasets using these color volumes when available and also for directly transferring from the 2D source images to more readily available scalar volumes (e.g., high resolution CT datasets) by considering the scalar volumes as grey-scale color volumes.

Our solution is composed of three steps: clustering, mapping, and transferring. We use the following two assumptions based on similarities between the example images and source volumes:

1. Simplicity: Illustrations can improve our understanding of the subject by omitting some unnecessary details. Therefore, we assume an illustration is a simplified drawing of the real object. Since illustrations usually employ different colors for different objects, we assume that if two objects do not have the same colors in the example, they will not share the same colors in the source. This assumption is unavoidable in an automatic method with no user interaction.

2. Similarity: Scientific illustrations often effectively capture object features, including relative area and volume propor-

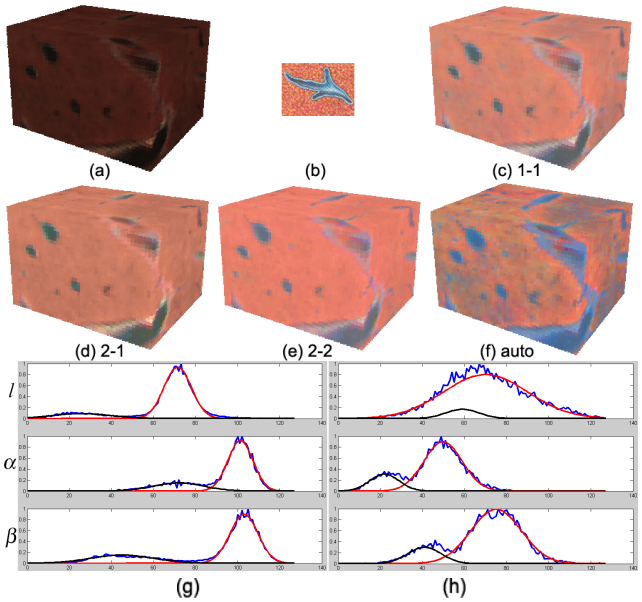

Figure 3: (a) A liver source volume. (b) An example cropped from the examples of Figure 10(b). (c)-(f) The color transfer results from (b) to (a) with different cluster numbers. (g)-(h) The clustering results of (a) and (b), while blue shows color histograms, and red and black show the clusters.

tions. Therefore, we assume that object distributions are similar between the sources and examples. This assumption is exhibited by most illustrations and we only need a rough correspondence for the mapping step.

3.2 Color Transfer Process

During the clustering step, we find color cluster sets using color distribution information from the example image and source volume, respectively. The color histograms are gathered from the whole texture for the 3 channels of the $l\alpha\beta$ color space [22]. We use this color space because of the independence between the luminance and two chrominance channels. Since the color transfer is based on the means and variances of the clusters, a set of Gaussian functions is fit onto the color histograms. These functions correspond to the color clusters in the textures. For example, Figure 3(g) and (h) show the clustering results for the liver volume (a) and an illustration example (b), in which the red line corresponds to the liver body and the black line corresponds to the veins and arteries. Therefore, each function must have the same area on the three color distributions, although different shapes are possible. The best parameters for the set of quantized Gaussian functions (center locations, weights, and heights) are searched under this area restriction. The algorithm starts with the set of one function and stops when the maximum error is smaller than a user-specified threshold (e.g., 5%). Based on the first assumption, the color distribution of the source has at least as many clusters as the example. Therefore, we first fit the example image, then the source volume until both the error threshold is satisfied and the number of clusters is greater than or equal to that of the example. For the clustering process, the color histograms are scaled from 0 to 1.

The mapping step finds a correspondence map from the source color clusters $G_s = \{G_{si},\ i = 1 \cdots N_s\}$ to the example color clusters $G_e = \{G_{ej},\ j = 1 \cdots N_e\}$. Based on assumption one, each source cluster only corresponds to one example cluster; therefore, $N_s \geq N_e$ and there exists such a multiple to one mapping from G_s to G_e. Based on assumption two, this mapping should satisfy similar object distributions between the examples and sources. Assuming $S = \{s_i = Area(G_{si})\}$ and $E = \{e_j = Area(G_{ej})\}$ are the normalized area sets of the color clusters of the source and example, respectively. The mapping problem can be described as finding a multiple-to-one mapping $f(S) \rightarrow E$ for the minimization of the following mapping error:

$$M_{Error} = \sum_{i=1}^{N_s} (s_i - f(s_i))^2,\ where \sum_{i=1}^{N_s} s_i = 1,\ \sum_{j=1}^{N_e} e_j = 1,\ and\ N_s \geq N_e. \quad (1)$$

When $N_s = N_e$, this map is a one to one correspondence. The mapping errors from the N_s factorial combinations are calculated to find a solution with the minimum error. Because of the algorithm complexity, a greedy algorithm can be used to produce a relative optimization solution. The two cluster sets S and E are sorted separately, and then the two clusters are mapped from the source to the example directly if they share the same sequence in their clusters, as $f(s_i') = i$. The mappings are designed randomly if multiple items with the same value exist, which rarely happens with real data.

When $N_s > N_e$, the map is a multiple to one correspondence, which means the examples omit some of the details by merging multiple real objects into the same region. We generate intermediate source cluster set S' with size N_e by merging clusters together. Then, we use the $N_s = N_e$ case to find the mapping error for each intermediate cluster set, and choose the mapping f' with the minimum mapping error. The mapping function f can be easily built from f' by reversing the merging sequence.

After building the correspondence map, we perform the color transfer on the three $l\alpha\beta$ channels at the same time, since they are combined in the color cluster sets. For each voxel $\vec{c_v}$ in the source volume, we first calculate the distance $\vec{d_i}$ from $\vec{c_v}$ to each center of the source color clusters $\vec{E_{si}}$ in 3D. The norm D_i is calcuated by using the value ranges $\vec{V}$ to balance the three channels. Then, we calculate the transferred color $\vec{c_i}$ to every source cluster from the corresponding example cluster. The final transferred color vector $\vec{C}$ is calculated by using an inverse function $q()$ of the distance D_i.

$$\vec{C} = \frac{\sum_{i=1}^{N_s} (\vec{c_i} * q(D_i))}{\sum_{i=1}^{N_s} (q(D_i))},\ where\ D_i = \sqrt{\sum_{j=l,\alpha,\beta} (d_{i,j}/V_j)} \quad (2)$$

$$and\ c_{i,j} = \frac{(c_{v,j} - E_{si,j}) * (\sigma_{ef(s_i),j})}{\sigma_{si,j}} + E_{ef(s_i),j},\ j = l, \alpha, \beta \quad (3)$$

Since scalar volumes only have a luminance distribution, the cluster set of the source is searched using this channel, and the mapping is still built based on cluster area proportions. During the transfer step, D_i is calculated as $|c_v - E_{si}|$. The color vector $\vec{c_v}$ is composed by the source scalar value as $[c_v, c_v, c_v]$ and so are the mean $\vec{E_{si}}$ and deviation $\vec{\sigma_{si}}$. The final transferred color $\vec{C}$ is calculated using the rest of the equations (2) and (3) as for a colored volume.

3.3 Color Transfer Results and Discussions

Figure 3 (c)-(f) show the color transfer results from (b) to (a) with different source and example cluster numbers. Since (e) and (f) save some color blending by transferring colors between the corresponding regions, they produce more vivid colors than (c, d). The result (e) is dependent on the matching of the correspondences, which requires user interaction to achieve a satisfying result. With an automatic process to detect the clusters from color distributions, the cluster numbers and parameters can be calculated more accurately; thereby producing a natural color range in (f) which matches the average color tone of the examples. Using the two assumptions

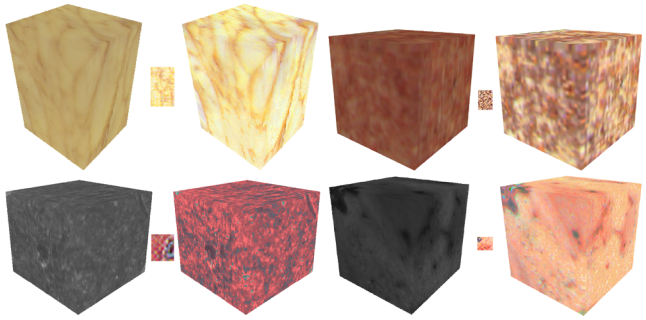

Figure 4: The sources, examples, and color transfer results of colored volumes (top: fat and vessel) and scalar volumes (bottom: spleen and kidney).

based on the similarity between examples and sources, we are able to build the correspondence between the clusters automatically to save user interaction. As the 4 other results for colored and scalar volumes shown in Figure 4 illustrate, this method can be used to generate 3D textures for a wide range of examples and source volumes, including artificial 3D textures [14]. Figure 9 and 10 demonstrate 13 results (6 for the hands and 7 for the abdomen) on 10 different objects. Although we only transfer colors between one example and one source, multiple examples or sources will also work since our method only needs the color distributions.

While texture synthesis methods may produce more continuous textures, they are not practical for small examples. We choose to use a color transfer approach instead of texture synthesis because of the similarities between scientific illustrations and real objects (Figure 2), and these results also demonstrate the effectiveness of illustrations capturing at the subject features. This approach may be used for other types of source images, with the requirement of the similarities between the sources and examples. However, the two assumptions impose some limits on their application. For instance, if the blood vessels in CT data have the same values, the color transfer process works, but cannot separate veins from arteries automatically.

4 Texture Synthesis for Wang Cubes

The requirements of large memory space and long synthesis time for high-resolution 3D textures limit their use in volume applications. Therefore, we use Wang Cubes, the 3D extension of 2D Wang Tiles, to overcome both issues and create non-periodic textures for our illustrative renderings. In this section, we present an automatic cube synthesis method from a 3D sample volume, discuss an edge discontinuity problem with Wang Cubes, and give two methods to solve the problem.

4.1 Automatic Cube Synthesis

Wang Tiles are square tiles with "colored" edges. They are placed on a plane edge-to-edge only if the adjacent edges share the same "color". Similarly, Wang Cubes are cubes with "colored" faces in the sense that two cubes can be put together only if the adjacent faces have matching "colors." Let's denote the cube faces as N, S, W, E, F, and B, as shown in Figure 5(h). Since Wang Cubes are not supposed to be rotated, two faces in the same direction (NS, WE, and FB) must share one set of colors, while the face colors on different directions are independent. Figure 5(a)-(d) show the usage of a 16 Wang Cube set. (a) shows a set of Wang Cubes with colored faces and 2 colors for each NS, WE, FB direction. (b) is a 8^3 tiling generated with the Wang Cube set in (a). We render each

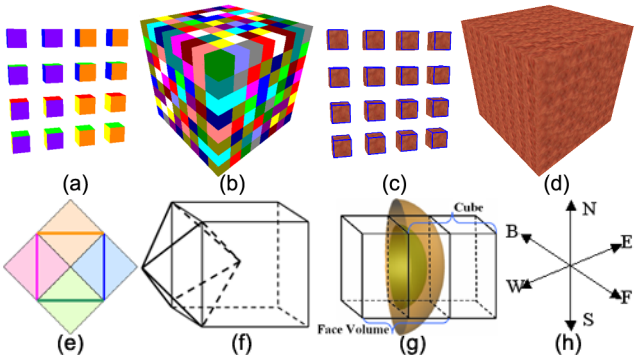

Figure 5: (a) A set of 16 Wang Cubes with 2 colors on each face. (b) A 8^3 cube tiling with each cube a color. (c) A set of synthesized cubes. (d) A composed 3D texture. (e) Tile generation using diamond samples. (f) Cube generation using an octahedron for the W cube face. (g) Cube generation using "face volumes" with two elliptical spheroids to restrict the synthesis shape for the W cube face. (h) The cube face directions.

cube with a color to show the tiling is non-periodic. (c) shows the same set of Wang Cubes (a) filled with 3D textures and (d) shows the composed larger volume texture by putting the cube contents (c) into the corresponding positions of the cube tiling in (b). The essential point of using Wang Cubes is that the cube face colors are only used to generate the non-periodic tilings, while the contents of the cubes can be filled with arbitrary textures. Once the cube contents are generated, new non-periodic 3D textures can be quickly generated using these cube tilings.

For 2D Wang Tiles, Cohen et al. [4] construct each tile by finding cutting paths to combine the four sample diamonds that correspond to the edge colors of the tile, as shown in Figure 5(e). Sibley et al. [24] and Lu et al. [16] extend the 2D tile generation to 3D cubes by using an octahedron to correspond to a face color and synthesizing a cube with 6 octahedra (Figure 5(f)). Four different corner-to-corner synthesis processes are needed to distinguish the synthesis directions during the cube generation. Here, we make the following modifications to simplify this method. Replacing the octrahedra, a "face volume" is randomly chosen from the sample volume and corresponds to a cube face color. The face volumes have the same size as the Wang Cubes and are used in a similar way as the octahedra to compose a cube. Similar to the generation of 2D tiles, the usage of the face volumes changes the synthesis problem along the cube faces into synthesis inside each cube. Since the cutting surfaces inside a cube do not need to follow a special synthesis shape, two half elliptical spheroids are used to restrict the quilting regions between an initially randomly selected cube and a face volume for each cube face. Figure 5(g) shows this process for the W face of a cube. As in Sibley et al. [24] and Wang and Mueller [31], we adopt the graph cut algorithm which uses max-flow (min-cut) [2] for 3D texture synthesis. This process is repeated for each face of every cube until the accumulated synthesis errors E_i along all the cutting surfaces inside the cubes are below a desired threshold.

4.2 Edge Discontinuity Problem

However, both direct and simplified cube generation methods have an edge discontinuity problem, where the textures are discontinuous on the cube edges along the cube faces. As the $2 \times 2 \times 1$ cube tiling in Figure 6(a) shows, the adjacent faces of the four cubes share the same colors and, therefore, the same textures. Since A(E) and C(E) come from two randomly selected samples, they are not necessarily continuous at the edge $a1$ along this cube face. Figure 6(b) shows that this problem generates a texture with obvious discontinuities on

all the cube edges, although the textures are continuous inside each cube and within the adjacent faces. Therefore, considering only the face colors is not sufficient to generate an everywhere continuous 3D texture for Wang Cubes.

For the synthesis of general Wang Cube sets, we add an additional modification phase for the face volumes to ensure edge continuity. Since the 6 faces of a cube are independently designed, an edge in the cube tiling may be adjacent to any face color in the two vertical directions. When the joints of 2 arbitrary faces are not continuous, the cube tiling cannot guarantee edge continuities. Because of this face independence, the cube edges must be made the same in each direction to assure edge continuity. Therefore, for each of the NS, WE, and FB directions, an "edge volume" is randomly chosen from the sample volume and tied to this direction. For the face volumes that correspond to the colors in the WE directions, the NS and FB edge volumes are used to modify the 4 edge regions of the WE middle plane. Figure 6(c) shows the NS edge volumes for this case. Similar to the synthesis of a cube with face volumes, a quilting surface is searched within the two half cylinders between the face and edge volumes. The face volumes corresponding to the colors on the NS and FB directions are synthesized with WE-FB and NS-WE edge volumes respectively. The newly generated face volumes are used to generate the cubes using the previous method, thereby guaranteeing texture continuity.

An alternative method for textures without sharp boundaries is to introduce an edge error factor into the synthesis process. The accumulated synthesis errors, E_s, for a cube set are the weighted sum of the inside errors, E_i, decided previously and the newly added edge errors, E_e: $E_s = p_i \times E_i + p_e \times E_e$. E_e is used to measure the average edge discontinuties for the composed textures. We assume each cube is equally randomly picked during the cube tiling generation. Since the discontinuities along an edge come from the 2 vertical pairs of adjacent cube faces, for example, A(E)-C(E) and C(N)-D(N) for the edge $a1-b2-c3-d4$ in Figure 6(a). For each face of a cube, the errors of the 4 edges are calculated from all the cubes which have the same color on the adjacent face, so that the 2 face errors of the 12 edges from all the combinations are collected once and only once. To balance the inside and edge errors, their weights p_i and p_e are set as the inverse of their cutting path size. This simple modification works well for many example textures.

Figure 6 shows two synthesized volumes of a liver (d) and a heart (e) with a set of 16^3 cubes in a 8^3 cube tiling. The liver uses the extra edge volume fixing phase and the heart uses the edge error factor approach. Both methods take around 10 to 20 minutes to generate a 16 cube set with 16^3 cubes, while arbitrary sized new 3D textures are composed with a few seconds.

Instead of faces, the cubes can also be colored by corners. We can choose "corner volumes" that correspond to the corner colors and synthesize a cube with them in a similar way as the "face volumes." While this ensures texture continuity for the whole space, the minimum cube set will be increased to 128 cubes [16].

The tradeoff of using Wang Cubes to save texture memory is the repetitive appearance of the synthesized textures. Since a set of Wang Cubes has only a limited number of samples, it is inevitable that these samples will repeat themselves somewhere in the volume. This issue is also discussed for Wang Tiles in [4]. Therefore, Wang Cubes are especially useful to generate textures with similarities because of their non-periodic tiling. The synthesized volumes using Wang Cubes are not limited to homogeneous textures [16], and this repetitive appearance issue can be improved by larger cube sets.

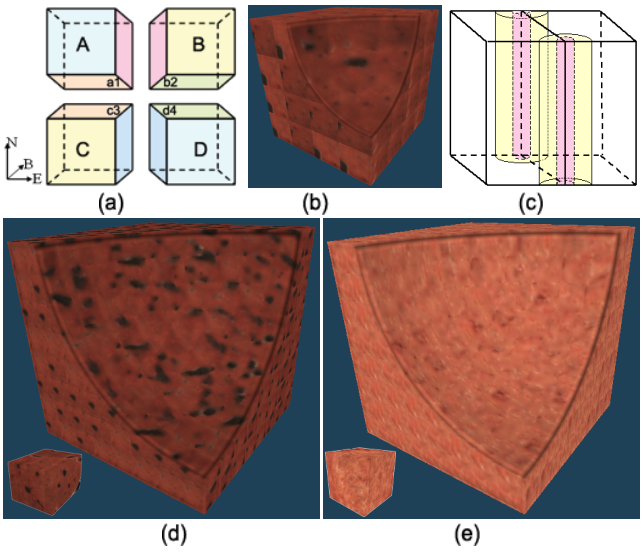

Figure 6: (a) A 2x2 cube tiling. (b) A result showing the edge discontinuity problem along the cube faces. (c) Edge volume usage for the FB edges of a WE face volume. (d) A liver texture generated using edge volumes. (e) A heart texture generated using the edge error factor.

5 Volume Applications

Simulating scientific illustrations, we have developed two volume applications that use our synthesized non-periodic 3D textures to enrich the details of the original datasets: a 2D slice viewer and a direct volume rendering system. The systems basically use the voxel values in the volumes to be visualized to provide the shape and opacity for each material and object. The user-chosen example illustration provides the colors and textures to generate three-dimensional textures used for rendering. To generate these detail textures for each object, a high-resolution 3D scalar or color volume corresponding to a representative real object is chosen. For segmented volume datasets, the segmentation masks containing one object ID for each voxel can be used to specify the texture index. For unsegmented datasets, we can select the desired texel value using transfer functions to generate voxel opacities that are then blended with the texel values. Moreover, transfer functions can always be used to modify the shape and opacity of an object for both segmented and unsegmented datasets.

Once the illustrative detail textures are generated, they are used for visualizing any scalar volume containing similar objects (e.g., abdominal CT scans for all patients). The rendering style is mainly determined by these color transferred textures, but can be modified by several selected rendering parameters and transfer functions. For example, using interactive 2D transfer functions, we usually make the skin transparent, so that both the skin and the volume interior can easily be seen. Comparing our method with traditional volume visualization, the illustrative 3D textures are used to color the scalar volume, replacing the traditional color transfer function. Therefore, our method preserves the fidelity of the original datasets and provides informative additional detail. The design of both applications preserves the flexibility of the direct volume rendering approach to interactively select which objects in a volume are visualized through simple user interaction.

Our sample texture volumes are chosen from high resolution volume datasets, such as the full colored Visible Woman photographic and CT datasets [1]. The datasets we used in Figure 7, 9, and 10 are segmented CT scalar volumes. Figure 8(b) is an unsegmented CT feet dataset. The selection of 3D texture samples proceeds by automatically searching the largest sample volume from the whole dataset, matching several standard statistical texture features with the user-selected sample images. The 3D texture samples can then be used directly to synthesize sets of cubes (producing rendered

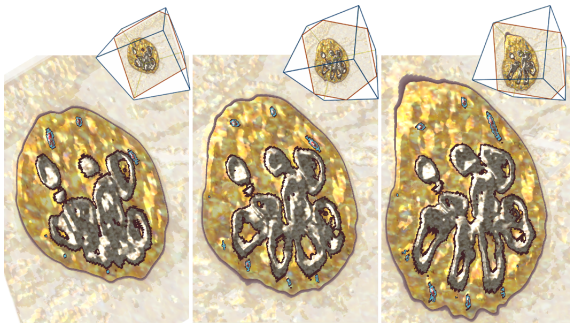

Figure 7: A series of hand slices as the cutting plane moves through the volume.

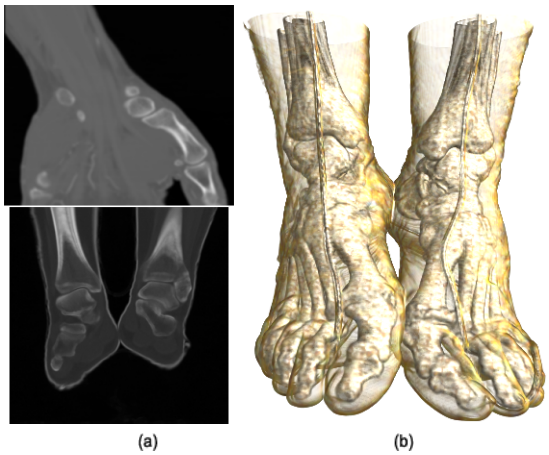

Figure 8: (a) Two CT slices of the hand and feet show the limited resolution of the original datasets. (b) Volume illustration of the unsegmented feet dataset. The skin and bone textures reuse the cube textures generated for Figure 9(c).

texture emulating the high resolution source volume), or after color is transferred from 2D example illustrations. We use one minimum Wang Cube set with 16 cubes for all the textures. Since all of them have the same cube face color set, one cube tiling for the whole volume is shared by all the synthesized cube sets.

5.1 2D Slice Viewer

The 2D slice viewer uses the synthesized 3D textures and segmentation masks to generate illustrative slices. From the object ID of the segmentation masks, the pixel color is fetched from the corresponding composed 3D textures by using the transformed world position. Although 2D images can be generated with Wang Tiles or other 2D texture synthesis methods, extra operations, such as smoothing adjacent images, are needed to avoid jumping and popping effects when the user moves through the slices. 3D textures provide smooth transitions naturally and the slice can be rotated into any view direction at any position in the volume. Gradient information from the dataset can also be used in the lighting to create a relief texture effect. These illustrative slices are generated and rendered in real-time. Figure 7 shows a series of hand slice images as the slice moves through the volume.

5.2 Direct Volume Rendering

We also built a GPU-based volume rendering system (slicing) to directly illustrate volume datasets. Since the cube tiling has the same size as the volume data, it is combined with the volume data and segmentation masks into one texture unit to save texture memory. Assuming M sets of cubes are used and C is the length of the cube side, all the cube textures are grouped into a grid of $4C \times 4C \times MC$, where each cube is still stored as a volume so that tri-linear interpolation can be achieved. One problem that can occur is that the details from the composed textures may be too small to be seen if we only allow each cube to map onto one voxel. Therefore, we add a rendering scale parameter for each cube set, which corresponds to the voxel numbers in each direction. Additionally, some textures might not be seen clearly after rendering, mainly because of the lack of texture variations from the sample volumes and the usage of specific rendering settings. We sharpen the cube textures while maintaining the average colors to overcome this problem.

The main difficulty in implementation is the calculation of texture coordinates in the fragment program. Since the composed 3D textures (scaled by the rendering scale parameter) are stored separately as the synthesized cubes and a cube tiling, we need to calculate the transformed coordinates for each fragment. Also, since the cubes for all the textures are packed into one texture unit, the texture coordinates must be accurately mapped inside a cube instead of between the cubes. For each fragment, we calculate two positions from the world coordinates: the voxel center position and the relative shift of the current world position to the left-down-back corner of the voxel. Then, the current segmentation mask is used to retrieve the render scale for this object. The voxel center position and the render scale are used to calculate the cube index from the cube tiling. Finally, the relative position and the render scale are used to retrieve the color of the current position in the cube.

After retrieving the cube color, we can use silhouette enhancement and lighting effects [9], which are commonly used in illustrations, in addition to transfer functions to calculate the final color for the current world position. Figure 9(a) and 10(a) are generated from the full colored slices of the visible human by synthesizing cube textures directly from the input textures. Figure 9(b, c) and 10(b) are illustrations with color transferred textures from the corresponding illustrative examples. All the results share the same data resolution, but differ in style. To generate the image in Figure 9(c), an elliptical spheroid is used to gradually cut away the fat, while parallel cutting planes are used in generating Figure 10(b). The two examples of the hand come from [25] and the abdomens are from [27]. The rendering time for these final, high-resolution renderings is around 1 second per frame for 500x500 images and 500 slices on a Nvidia Quadro FX 3400 graphics card.

The illustration in Figure 1 includes more object details than the right MR image although they have the same size. Similarly, in our rendering, the source volume data has limited information, while we generate cube textures from high-resolution data and use them to enrich the previous volume data. To provide meaningful information, we usually choose corresponding cube textures for each object/region in a volume. For example, we choose liver textures to render the liver, while skin textures are used to render skin. These corresponding textures contain the information about object appearance, color, component, and shape and can be used to improve the understanding of the data for education and training.

To further simulate the example illustrations, we interactively adjust the rendering parameters. For instance, we add silhouettes in the volume rendering if there are silhouettes in the example, and we use more transparent skin in the volume rendering if the skin region is not the focus point in the example. In our direct volume rendering approach, the users can interactively adjust the lighting direction, transfer functions for opacities, and four rendering parameters (1 for render scale, 2 for lighting, and 1 for silhouette) per object. Figure 8, 9, and 10 have 2, 3, and 8 objects respectively. After the cubes are generated, it takes a skilled user a few minutes to visualize an unknown dataset.

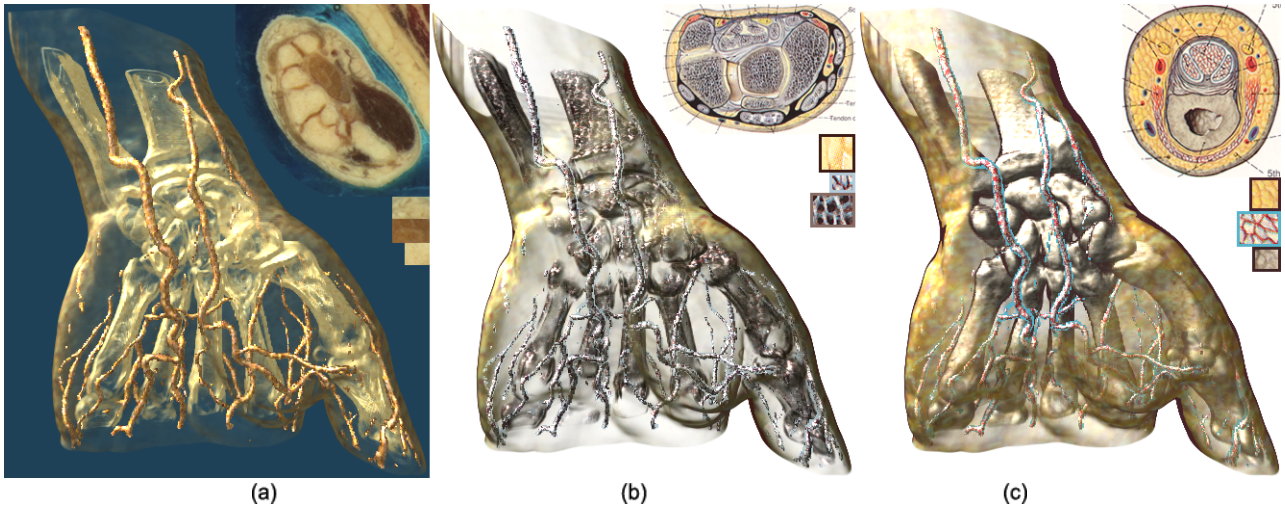

Figure 9: Volume rendering of a hand dataset. (a) is rendered with the cubes synthesized from the Visible Woman photographic dataset. (b) and (c) are rendered with recolored textures using their respective example illustrations. The three small images are cropped from the corresponding example regions and show the textures of fat, vessel, and bone, respectively. The silhouette colors in (b) and (c) are also selected from the examples and shown as the box colors around the samples.

5.3 User Interaction and Storage Requirements

Because of the ability to emulate examples by transferring their color distributions, our method requires much less user interaction than traditional rendering approaches. Only the 2D illustration and sample examples need to be selected by the user and the 3D textures are automatically generated. Our approach can render both segmented and un-segmented datasets; therefore, most user interaction occurs when adjusting the transfer functions and limited rendering parameters. Moreover, once the cube set is synthesized, it can be used to generate textures for the same or similar subject of any size and the cube sets are usually reusable. For example, the skin, vessel, and bone textures generated for the hand data (Figure 9) can be used to render the feet data (Figure 8(b)). What's more, once the cube set is synthesized, arbitrary sized 3D textures can be composed quickly.

Another advantage of Wang Cubes for volume rendering is that these non-periodic textures occupy much less texture space. Since the cube tiling is combined with the volume data and segmentation masks into one texture unit, it does not use an extra texture unit. The main memory difference is for storing the 3D textures. With Wang Cubes, the texture space of M sets of 16 cubes is $M \times 16 \times C^3$. To achieve the same effect of non-periodic textures without Wang Cubes, M volumetric textures having the same size of the dataset are needed using $M \times V^3$ space. Therefore, the difference of the space requirement is $16C^3/V^3$. In Figure 9 and 10, both datasets are $256 \times 256 \times 128$ and the cubes are 16^3. Therefore, the required texture memory with Wang Cubes is only 1/128 of that required without Wang Cubes.

6 Discussion and Future Work

This paper employs 3D textures to improve the visualization of volume datasets by enriching meaningful details and achieving illustrative styles. Based on the methods from scientific illustrations for conveying object features, we have built an automatic process to simulate illustration "rendering styles" using color transfer and illustration similarities. These "rendering styles" capture important illustration information, such as accepted conventions and usage of colors. By simulating these styles, the proposed method can generate expressive and pleasing rendering results with a much simpler process than by manual adjusting all parameters and colormaps.

To synthesize 3D textures from 2D examples is a very challenging task. By carefully observing the features of scientific illustrations, we use color transfer methods and available volume datasets to automatically generate illustrative 3D textures, which can generate high resolution 3D textures from small 2D examples with little user efforts. The usage of Wang Cubes significantly alleviates the issue of limited texture space. These synthesized 3D textures can be used in both surface-based methods and direct volume rendering. Our new rendering method keeps the advantage of direct volume rendering that a user can arbitrarily choose the object/region shape with transfer functions and change their focus of intent.

Since many medical anatomy textbooks and atlases uses illustrations in addition to medical images (such as CT or MR), we believe that this method is useful for educational applications. This method can be extended to several areas of 3D texturing and color transfer, including video synthesis and texture generation for other types of models.

We plan to further analyze and simulate the abilities of scientific illustrations to extract important object features and render volumetric datasets in a coherent and artistic way, such as highly-structured textures and illustrative lighting effects. Since the subtle effects of lighting and color usage around different portions of an object, some examples have inhomogeneous properties that affect the texture synthesis results. We plan to investigate new methods to remove or balance these issues. We also plan to explore artistic and scientific composition principles for adjusting rendering parameters to achieve the desired effects with much less user interactions.

Since texture memory size is a limited resource, we will further explore texture usage for volume rendering to provide more rendering styles for different applications, while accelerating the rendering speed by avoiding irrelevant texture memory accesses and texture coordinate manipulation.

7 Acknowledgement

The datasets are courtesy of Visible Human Project and Tiani Medgraph. We would like to thank Lujin Wang for texture synthesis and Jingshu Huang for Gaussian fitting. This project is supported by the National Science Foundation under Grant Nos. 0081581, 0121288, 0222675, 0328984, 9978032.

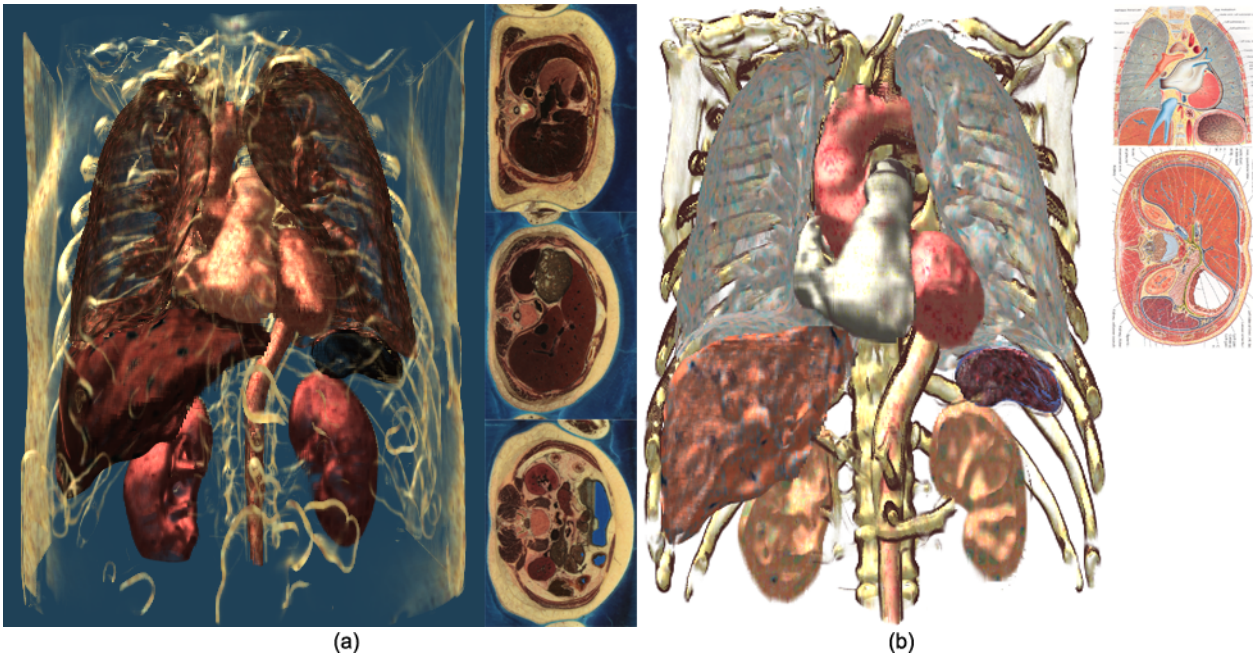

Figure 10: Volume rendering of an abdomen dataset. (a) is rendered with the cubes synthesized from the Visible Woman photographic dataset. The three slices on the right show the lung, liver, and kidneys respectively. (b) is rendered with recolored textures using the corresponding examples from the two right illustrations.

REFERENCES

[1] M. Ackerman. The visible human project. *Proceedings of the IEEE*, 86(3):504–511, 1998.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *IEEE Trans. on PAMI*, 26(9):1124–1137, 2004.

[3] Y. Chen, X. Tong, S. Lin, J. Wang, B. Guo, and H. Shum. Shell texture functions. In *SIGGRAPH*, pages 343–353, 2004.

[4] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *SIGGRAPH*, pages 287–294, 2003.

[5] Karel Culik II and Jarkko Kari. An aperiodic set of wang cubes. *J.UCS*, 1(10):675–686, 1995.

[6] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. Example-based style synthesis. In *IEEE CVPR*, 2003.

[7] D. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *IEEE Vis*, pages 195–202, 2000.

[8] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *Intl. J. Computer Vision*, 40(1):25–47, 2000.

[9] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *IEEE Visualization 2003*, pages 301–308, 2003.

[10] J. Hamel and T. Strothotte. Capturing and re-using rendition styles for non-photorealistic rendering. *CGF*, 18(3):173–182, 1999.

[11] A. Haro and I. Essa. Exemplar based surface texture. In *VMV 2003*, 2003. Munich, Germany, November 2003.

[12] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *PROC. of SIGGRAPH*, pages 327–340, 2001.

[13] E. Hodges. *The Guild Handbook of Scientific Illustration*. John Wiley and Sons, 1989.

[14] R. Jagnow, J. Dorsey, and H. Rushmeier. Stereological techniques for solid textures. In *PROC. of SIGGRAPH 2004*, pages 329–335, 2004.

[15] R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2d incompressible flows using concepts from painting. In *IEEE Visualization*, pages 333–340, 1999.

[16] A. Lu, D. Ebert, W. Qiao, M. Kraus, and B. Mora. Volume illustration using wang cubes. Number TR-ECE-04-05, 2004. https://engineering.purdue.edu/ECE/Research/TR/TR/2004.whtml.

[17] E. Lum and K. L Ma. Hardware-accelerated parallel non-photoralistic volume rendering. In *PROC. of NPAR*, pages 67–74, 2002.

[18] Zoltan Nagy, Jens Schneider, and Rdiger Westermann. Interactive volume illustration. In *VMV*, pages 497–504, 2002.

[19] Fabrice Neyret and Marie-Paule Cani. Pattern-based texturing revisited. In *SIGGRAPH 1999*, pages 235–242, 1999.

[20] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi. Volume illustration: Designing 3d models with internal textures. In *PROC. of SIGGRAPH 2004*, pages 322–328, 2004.

[21] E. Reinhard, Gooch B. Ashikhmin, M., and Shirley P. Color transfer between images. *IEEE CG&A*, pages 34–40, 2001.

[22] D. L. Ruderman, T. W. Cronin, and C. C. Chiao. Statistics of cone responses to natural images: Implications for visual coding. *J. Optical Soc. Of America*, 15(8):2306–2045, 1997.

[23] T. Saito. Real-time previewing for volume visualization. In *Proc. of symposium on Volume visualization*, pages 99–106, 1994.

[24] P. G. Sibley, P. Montgomery, and G. E. Marai. Wang cubes for video synthesis and geometry placement, 2004. SIGGRAPH Poster.

[25] F. J. Slaby, S. K. McCune, and R. W. Summers. *Gross Anatomy in the practice of medicine*. Lea and Febiger, 1994.

[26] J. Stam. Aperiodic texture mapping. Tech. Rep. R046, European Research Consortium for Informatics and Mathematics, Jan 1997.

[27] J. Staubesand and Anna N. Taylor(Editor). *Sobotta Atlas of human anatomy.* Urban and Schwarzenberg Baltimore-Munich, 1990.

[28] S. M. F. Treavett, M. Chen, R. Satherley, and M. W. Jones. Volumes of expression: Artistic modelling and rendering of volume datasets. In *Computer Graphics International 2001*, pages 99–106, July 2001.

[29] H. Wang. Proving theorems by pattern recognition ii. *Bell Systems Technical Journal*, 40:1–42, 1961.

[30] H. Wang. Games, logic, and computers. *Scientific American*, pages 98–106, November 1965.

[31] L. Wang and K. Mueller. Generating sub-resolution detail in image and volumes using constrained texture synthesis. In *IEEE Visualization*, pages 75–82, 2004.

[32] L. Y Wei. *Texture synthesis by fixed neighborhood searching*. PhD thesis, Stanford University, 2001.

[33] W. Welse, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. In *PROC. of SIGGRAPH*, pages 277–280, 2002.